Multi - Class Sentimental Analysis

Instructor: Dr. T. Akilan
Professor at Lakehead University

Harshil Chetankumar Shah
Student ID: 1111677
Lakehead University
Email id: hshah24@lakeheadu.ca

***Abstract*—The paper describes a multi-class sentimental analysis using a neural network architecture of five layer (Dense, Dropout, Flatten, MaxPooling, Conv1D). The aim of the project is to experiment with machine learning algorithm to predict the sentiment of the unseen review from the corpus that has the additional sentiments information of all sub-phrases. The main advantage of this model proposed is it justifies a problem of multi-class sentimental analysis based on Rotten Tomatoes movie review taking 70 percent of data for training and 30 percent of data for testing for the model.The Rotten Tomatoes movie review corpus has been greatly improved upon and annotated with a fine sentiment score. We want to see whether having finer sentiment annotations for every span of parsed sentence in the training set would help improve the accuracy in predicting the overall sentiment of the unseen sentence. After performing the analysis, the model have achieved an accuracy of , loss of ,f1-sscore of .**

***Index Terms*—Convolutional layer, normalization, epochs, ReLU(Rectified Linear Unit), softmax**

I. Introduction

In today's era, text classification plays an important role, where the information available is devastating. The text classification work is related to topical classification, the categorization of more subjective documents that depend more on style and the author's opinion is also important. Many of the websites such as Rotten tomatoes, IMDB, rely on the opinions and reviews to keep their sites active. With the increasing ubiquity of social media such as Twitter and review sites like Yelp and Rotten Tomatoes, it is important to be able to automatically make sense of these large amounts of subjective data.

A. What is Sentimental Analysis ?

Sentiment analysis, using natural language processing techniques is used to characterize subjective human opinions or sentiments, has been rapidly gaining popularity as a method of analyzing these large corpora for such diverse applications.

Sentiment analysis is the computational task of automatically determining what feelings a user is expressing in a text or a comment. Sentiment is often framed as binary classification or multi-class classification. In binary classification, the positive and negative review can be refrained from the movie review corpus. Whereas, in multi-class classification, more than one classification such as, negative, somewhat negative, neutral, somewhat positive, positive etc. is made on the sentiments from the review corpus.

Sentimental analysis is used in various applications such as stock predicting, movie review etc. During the Obama administration, sentiment analysis was used to gauge a public opinion to policy announcements and campaign messages before 2012 presidential election. The sentimental analysis is the integral part of the market research and customer service approach. The use of sentimental analysis is to quickly brief some qualities of the text, especially when the text is in large quantity and a human cannot analyze it.

B. Application of Sentimental Analysis

1) Applications to Review Websites : Movie Reviews, Product Reviews etc.

2) Applications as a Sub-Component Technology : Spam detection, Detecting antagonistic, heated language in mails, context sensitive information detection etc.

3) Applications in Business : Knowing Customer attitudes and trends

4) Applications across Different Domains : Knowing public opinions for political leaders or their notions about rules and regulations in place etc.

II. Literature Review

Most of the work in text classification has been topical classification. However, there has also been work in classifying text based on issues beyond subject matter, such as author identification and sentiment classification. Previous work in sentiment classification has mostly focused on classifying reviews as positive and negative. Pang, Lee, and Vaithyanathan have used learning algorithms such as Naive Bayes, SVM, and maximum entropy to classify reviews; however, their work focused on considering on the presence of terms (rather than their frequency) and using only a limited number of features. Pang and Lee have also gone further and worked on classifying reviews using a "multi-point" scale, such as the number of stars for a movie review. In addition, there also have been previous work [1] using unsupervised learning algorithms to classify reviews. Specific study on the task of multi-class classification of online posts of Twitter users, and show how far it is possible to go with the classification, and the limitations and difficulties of this task [2]. The paper "Sentiment analysis

TABLE I
DISPLAYING FIVE ROWS

	PhraseID	SentenceID	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage...	1
1	2	1	A series of escapades demonstrating the adage...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

TABLE II
NEW LABELS GENERATION

	PhraseID	SentenceID	Phrase	Sentiment
0	116889	6236	Sade set	2
1	54160	2690	gives a human face to what's often discussed...	3
2	103755	5470	ever saw that was written down	2
3	47276	2304	a breadthtakingly assured and stylish work	4
4	21710	970	the eventual DVD release will offer subtitles...	2

for Movie Reviews" by Ankit Goyal, and Amey Parulekar worked on the "Large Movie Review" dataset used by the AI department of Stanford University for associated publication. The dataset contained 50,000 training examples collected from IMDB where each movie review is labelled with rating on a scale of 1 to 10. They have categorized the ratings as either 1 (like) or 0 (dislike) based on the IMDB ratings. If the rating was above 5, it was deduced that the person liked the movie otherwise he did not. The Cross Validation was used in which the complete dataset was divided into multiple folds with different samples for training and validation each time and the final performance of the statistic of classifier was averaged over all results. The 3 methods used for feature extraction were Bag of words, N-gram modelling and TF-IDF modelling. They have tried multiple classification models on various feature representations of the textual information in the reviews. Out of all the models, the logistic regression model showed best performance [3].

III. MOVIE REVIEW DATASET

I am using movie reviews data from the https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset /master/train.tsv. The rotten tomatoes movie review dataset is a corpus of movie reviews used for the sentimental analysis, originally collected by the Pang and Lee. In their work on sentimental analysis, Socher et al have used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the database. The corpus is consisting of 156060 phrases. It has four columns that are PhraseID, SentencID, Phrase and Sentiment. The first five row entries of the dataset are displayed in Table I. Further, traditional test-train-split using random initialization from sklearn.model selection, here I am splitting the data into training and testing. The data is splitted into 70 percent for training and 30 percent for testing. Now, both the data (i.e., training and testing) are converted into a NumPy array using training and testing, because converting it into a NumPy makes the data easy for manipulation. The dataset is shuffled and randomly any five row entries are displayed as shown in Table II.

```
X_train, X_test, Y_train, Y_test = train_test_split(
    df ['Phrase'], df ['Sentiment'], test_size=0.3,
    random_state=2003)
documents=[]
X_train = np.array(X_train.values.tolist())
Y_train = np.array(Y_train.values.tolist())
for i in range(len(X_train)):
  documents.append([list(word_tokenize(X_train[i])),
     Y_train[i]])

X_test = np.array(X_test.values.tolist())
Y_test = np.array(Y_test.values.tolist())
for i in range(len(X_test)):
  documents.append([list(word_tokenize(X_test[i])),
    Y_test[i]])

documents[0]
```

Listing 1. Splitting of Data

A. Libraries

In order to do the classification using 1D-CNN, I have used the pandas library to load the data and do some pre-processing part. In order to do mathematical calculation or converting the pandas data-frame to array I have used numpy. Finally, in order to develop the CNN model I have used the Keras framework. So, below is the list of library used:

- pandas
- numpy
- seaborn
- matplotlib
- sklearn

```

from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Listing 2. Importing Libraries

IV. PREPROCESSING DATA

Pre-processing refers to the transformations applied to the data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. The following are the preprocessing steps for converting the raw data into the clean data:

1) Data Cleaning: : data cleaning, the punctuations, unnecessary character, numbers etc.

2) Stemming:: Stemming refers to the process of chopping of the end words making the word more sensible. So stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.

3) Lemmatization: : It reduces the inflected words properly ensuring that the root word belongs to the language. The root word is called as lemma. A lemma is a dictionary form or citation form of a set of words. Lemmatization is used in comprehensive system retrieval systems like search engines etc.

4) Stopwords: : In the text, there may be some stopwords like ‘the’, ‘is’, ‘are’ which need to be removed from the text because they are not so interesting. In English language there are 179 stopwords. The below in fig 1 is the flow of the preprocessing step for the raw data.

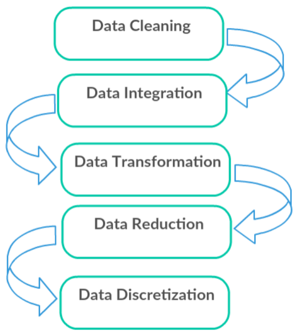

Fig. 1. Preprocessing Steps

```
for l in range(len(documents)):
  label = documents[l][1]
  tmpReview = []
  for w in documents[l][0]:
    newWord = w
    if remove_stopwords and (w in stopwords_en):
      continue
    if removePuncs and (w in punctuations):
      continue
    if useStemming:
      newWord = lancaster.stem(newWord)
    if useLemma:
      newWord = wordnet_lemmatizer.lemmatize(newWord)
    tmpReview.append(newWord)
  documents[l] = (tmpReview, label)
  documents[l] = (' '.join(tmpReview), label)

print(documents[0])
```

Listing 3. Preprocessing of Raw Data

V. Proposed Method

The model which I have proposed comprises of many different layers which makes the model more accurate. The very first layer is Convolution 1-D layer. In this layer, a convolution kernel is created that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, another convolution 1-D layer is added. The next layer is the MaxPooling layer. we reduce the size of the feature map. Max pooling is a sample-based discretization process. The objective is to down-sample an input representation , reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. The primary purpose of the convolution and pooling layer is the feature extraction.In the dropout layer, a function which randomly selects neurons are ignored during training. It is a simple way to prevent neural network from over fitting. The parameter rate in this layer depicts the value for dropout in the hidden layer whereby I have taken value as 0.5. The next layer is the Flatten layer. In this step, all the pooled feature maps are taken and put into a single vector. Its task is to convert the pooled feature map to a single column that is passed to the fully connected layer. The last layer for model training is the dense layer. In this layer, it adds the fully connected layer to the neural network. The first parameter is output dimension which is the number of nodes in the hidden layer and the second parameter is the activation function.

TABLE III
Testing Model

Test Loss	1.0260
Accuracy	0.6062
F-1 Score	0.5301
Precision	0.6538
Recall	0.5844

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
                 activation='relu',
                 input_shape=(2000,1)))
model.add(Conv1D(128, kernel_size=5, activation='relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(rate = 0.50))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))
```

Listing 4. Importing Layers in Keras

VI. Experimental Results

In the experimental analysis, I have tried several combinations of method with the hyperparameter. The different hyperparameter in the code includes: batch size, number of input layer, number of classes, random state, dropout rate, epochs, etc. While I noticed a change in the accuracy of the model by changing the input layers, epochs, learning rate, activation function, optimizer. The final model after testing is shown in the Table III

VII. Comparison between model

The primary goal is to achive a model with the best performance. Thus, in order to get a best training model, the model need to achieve a score near to 1. The differing accuracy may depend on the following factors: 1. Batch Size 2. Optimizer 3. Number of features. To different optimizer like AdaGrad, SGD, Adadelta were used in the model training. After implementing all the optimizer, the best test accuracy was found in the Adadelta optimizer, The different optimizer’s accuracy and loss function is in the Table IV

VIII. Conclusion

A model was trained using the optimizer Adadelta. The model has several layers for solving the problem of multi-class sentimental analysis using different activation functions

TABLE IV
DIFFERENT OPTIMIZER

Sr No.	Optimizer	Accuracy	Loss function	F1-Score
1	SGD	0.5877	1.0545	0.4590
2	AdaGrad	0.6054	1.0341	0.5190
3	Adadelta	0.6125	1.0221	0.5301

such as ReLU (Rectified Linear Unit) and softmax (mutli-class) or sigmoid (binary classification). Finally the sentiment analysis under the reference to Natural language Processing (NLP) determine whether a text contains some informative information and what particular information it depicts with respect to sentiments.

REFERENCES

[1] Bo Pang and Lillian Lee, Shivakumar Vaithyanathan, “Sentiment Classification using Machine Learning Techniques” , EMNLP Turneys’s 2002

[2] M. Bouazizi and T. Ohtsuki, ”Multi-class sentiment analysis on twitter: Classification performance and challenges,” in Big Data Mining and Analytics, vol. 2, no. 3, pp. 181-194, September 2019.

[3] Ankit Goyal, and Amey Parulekar, “Sentiment analysis for Movie Reviews”, Standford university, 2019